AMENDMENTS TO THE CLAIMS

Please cancel claim 4 without prejudice. All pending claims are listed below:

1. A method for optimizing a number of instructions in a program file, comprising:

finding a first instruction and a second instruction in the program file that each perform a single operation and the second instruction depends on a result of the first instruction; and

forming a fused instruction that includes

determining a fused opcode that represents both a first operation of the first instruction and a second operation of the second instruction.

2. The method of claim 1, wherein the second instruction depends on the result of the first instruction if a destination operand of the first instruction is at least one source operand of the second instruction.

3. The method of claim 1, further comprising:

replacing the first instruction and the second instruction with the fused instruction in the program file after the fused instruction is formed.

4. (Canceled)

5. The method of claim 1, wherein forming the fused instruction includes

combining at least one source operand of the first instruction and the at least one source operand of the second instruction that is not the destination operand of the first instruction to form a plurality of source operands of the fused instruction; and

setting a destination operand of the fused instruction to a destination operand of the second instruction.

6. The method of claim 5, wherein the first instruction has two source operands and the second instruction has two source operands, and the two source operands of the first instruction are combined with a particular one of the two source operands of the second instruction that is not the destination operand of the first instruction to form three source operands of the fused instruction.

7. The method of claim 1, wherein the fused opcode is specified using M * N bits and the fused opcode represents one of 2^{M} first operations fused with $2^{(M * N - M)}$ second operations.

8. The method of claim 6, wherein each of the three source operands of the fused instruction and the destination operand of the fused instruction are specified using a number of bits needed to address available registers.

9. A fused instruction, comprising:

a fused opcode that represents both a first operation of a first instruction and a second operation of a second instruction where the first instruction and the second instruction each perform a single operation and the second instruction depends on a result of the first instruction.

10. The fused instruction of claim 9, wherein the second instruction depends on the result of the first instruction if a destination operand of the first instruction is at least one source operand of the second instruction.

11. The fused instruction of claim 10, wherein the fused opcode swaps the order of applying a temporary operand, that stores the result of the first instruction, to the at least one source operand of the second instruction that is not the destination operand of the first instruction, if the second instruction is non-commutative and swaps the order of applying a first one of the at least one source operand of the second instruction to a second one of the at least one source operand of the second instruction.

12. The fused instruction of claim 10, further comprising:

a plurality of source operands formed by combining at least one source operand of the first instruction and the at least one source operand of the second instruction that is not the destination operand of the first instruction; and

a destination operand that is a destination operand of the second instruction.

13. The fused instruction of claim 12, wherein the plurality of source operands are three source operands that are formed by combining two source operands of the first instruction and a particular one of two source operands of the second instruction that is not the destination operand of the first instruction.

14. The fused instruction of claim 9, wherein the fused opcode is specified using M * N bits and the fused opcode represents one of 2^M first operations fused with $2^{(M * N - M)}$ second operations.

15. The fused instruction of claim 12, wherein each of the three source operands of the fused instruction and the destination operand of the fused instruction are specified using a number of bits needed to address available registers.

16. The fused instruction of claim 12, wherein the fused opcode is specified using eight bits and each of the three source operands of the fused instruction and the destination operand of the fused instruction is specified using five bits.

17. A compiler, comprising:

a fused instruction generator that

finds a first instruction and a second instruction in a program file that each perform a single operation and the second instruction depends on a result of the first instruction; and

forms a fused instruction that includes

a fused opcode that represents both a first operation of the first instruction and a second operation of the second instruction.

18. The compiler of claim 17, wherein the second instruction depends on the result of the first instruction if a destination operand of the first instruction is at least one source operand of the second instruction.

19. The compiler of claim 17, wherein the fused instruction generator replaces the first instruction and the second instruction with the fused instruction in the program file after forming the fused instruction.

20. The compiler of claim 18, wherein the fused opcode swaps the order of applying a temporary operand, that stores the result of the first instruction, to the at least one source operand of the second instruction that is not the destination operand of the first instruction, if the second instruction is non-commutative and swaps the order of applying a first one of the at least one source operand of the second instruction to a second one of the at least one source operand of the second instruction.

21. The compiler of claim 17, wherein the formed fused instruction includes

a plurality of source operands formed by combining at least one source operand of the first instruction and the at least one source operand of the second instruction that is not the destination operand of the first instruction; and

a destination operand that is a destination operand of the second instruction.

22. The compiler of claim 21, wherein the plurality of source operands are three source operands that are the two source operands of the first instruction and a particular one of the two source operands of the second instruction that is not the destination operand of the first instruction.

23. A processor, comprising:

a fused instruction execution unit that includes a first arithmetic logic unit (ALU) to perform a first operation and a second ALU to perform a second operation,

wherein a result of the first ALU is input into the second ALU and within one clock cycle, the first ALU performs the first operation and the second ALU performs the second operation.

24. The processor of claim 23, further comprising:

an instruction fetch/decode unit that tags a fused instruction for execution on the fused instruction execution unit, if the first ALU performs a first operation of the fused instruction and the second ALU performs a second operation of the fused instruction; and

a re-order unit, coupled to the instruction fetch/decode unit, that stores the fused instruction for retrieval by a reservation station.

25. The processor of claim 24, further comprising the reservation station, coupled to the re-order unit and the fused instruction execution unit, that retrieves the fused instruction and dispatches the fused instruction to the fused instruction execution unit, if the first operation of the first ALU matches the first operation of the fused instruction and

the second operation of the second ALU matches the second operation of the fused instruction.

26. A method for executing a fused instruction within one clock cycle, comprising:

performing a first operation in a first arithmetic logic unit (ALU) of a fused instruction execution unit and a second operation in a second ALU of the fused instruction execution unit,

wherein a result of the first ALU is input into the second ALU and within one clock cycle, the first ALU performs the first operation and the second ALU performs the second operation.

27. The method of claim 26, further comprising:

dispatching the fused instruction to the fused instruction execution unit if the first operation of the first ALU matches a first operation of the fused instruction and the second operation of the second ALU equals a second operation of the fused instruction.

28. A machine-readable medium having stored thereon data representing sequences of instructions, the sequences of instructions including sequence of instructions which, when executed by a processor, cause the processor to perform the steps of:

finding a first instruction and a second instruction in the program file that each perform a single operation and the second instruction depends on a result of the first instruction; and

forming a fused instruction that includes

determining a fused opcode that represents both a first operation of the first instruction and a second operation of the second instruction.

29. The machine-readable medium of claim 28, wherein the second instruction depends on the result of the first instruction if a destination operand of the first instruction is at least one source operand of the second instruction.